anneurai.net

Matlab-based IPython notebooks

I really like Python's philosophy, but over the last years I haven't been able to switch the code for my research from Matlab. At this point, the transition costs are too high for me, but it's a move I have planned for some point in the future.

Now, Python has the awesome Jupyter (formerly IPyton notebook) feature, that allows for comments, code, and most importantly graphical output (i.e. figures you've just generated) to be shown in one document. This is a great way to share and explain the code you're writing, since the reader immediately sees how output is generated without having to run all the analyses themselves.

Matlab has a Publish function that attempts to do something similar, and outputs HTML files containing both code and figures. However, there are a few drawbacks: it uses its own Markup text layout language, and GitHub (where I intend to host my notebooks) does not natively render html.

Luckily, some awesome people have made it really easy to generate IPython notebooks using Matlab, that are super easy to share or host on GitHub. Here's the step-by-step guide:

1. Download and install Anaconda https://www.continuum.io/downloads. Restart Terminal. Or, if you'd prefer to not get the full Anaconda software, check out this post.

2. In terminal, type
   ```
   pip install pymatbridge

   pip install matlab_kernel

   python -m matlab_kernel install
   ```

3. Point the kernel to your version of Matlab. I added `export MATLAB_EXECUTABLE=/Applications/MATLAB_R2015b.app/bin/matlab` to my *.bash_profile* file. To do this from Terminal, type `echo "export MATLAB_EXECUTABLE=/Applications/MATLAB_2015b.app/bin/matlab" >> ~/.bash_profile`. Of course, make sure the location and version of Matlab match

yours.

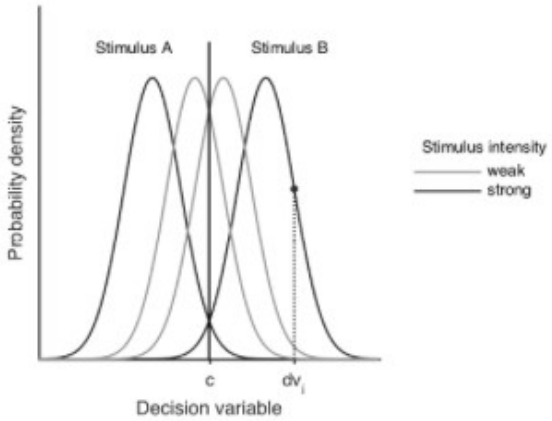
4. Restart terminal or load `.bash_profile`. Type `ipython notebook` in Terminal. Your browser will open a Jupyter window, where on the right hand side you can go to New -> Notebooks -> Matlab.

5. You're now ready to run your notebook! Simply write or copy a block of code from matlab, and click Run. The figures will automatically appear when you are calling any plotting function. When you push your Notebook.ipynb file to GitHub, it will automatically render the layout and figures. You can also download the whole notebook to HTML which can be viewed in any web browser.

See here for an example notebook, analysing some pupillometry data in Matlab. Here's a screenshot of a notebook on signal detection theory:

In [17]:

```
%% Show the basis of signal detection theory

% generate 4 distributions
stim = -0.6:0.001:0.6;
s1 = normpdf(stim, -.2, 0.1);
s2 = normpdf(stim,  .2, 0.1);
s3 = normpdf(stim, -.05, 0.1);
s4 = normpdf(stim,  .05, 0.1);

% plot those plus a criterion line in the middle
p = plot(stim, s1, stim, s2, stim, s3, stim, s4, zeros(1, 2), [0 4.5], 'k');

% two greys for different intensities of the stimulus
gr1 = [0.2 0.2 0.2];
gr2 = [0.6 0.6 0.6];

p(1).Color = gr1; p(2).Color = gr1;
p(3).Color = gr2; p(4).Color = gr2;

hold on; % one sample DV
plot(0.3, s2(dsearchn(stim', 0.3)), '.', ...
    'MarkerFaceColor', gr1, 'MarkerEdgeColor', gr1, 'MarkerSize', 10);
plot([0.3 0.3], [0 s2(dsearchn(stim', 0.3))], 'color', gr1, 'linestyle', ':');

% layout
axis tight; ylim([0 5]);
ylabel('Probability density'); xlabel('Decision variable');
text(-.4, 4.4, 'Stimulus A');
text(.15, 4.4, 'Stimulus B');

% add legend
l = legend([p(3), p(1)], 'weak', 'strong', 'Location', 'EastOutside');
l.Box = 'off';
text(0.75, 3, 'Stimulus intensity'); % legend title
set(gca, 'xtick', [0 0.3], 'xticklabel', {'c', 'dv_i'}, ...
    'box', 'off', 'tickdir', 'out', 'ytick', [], 'yticklabel', []);
```



This code works on a Mac OS X El Capitan, and requires some familiarity with Terminal.